**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №16

Сравнение лиц с использованием библиотеки face\_recognition

Выполнила: Студентка группы

БВТ2402

Шилинцева Татьяна

Москва

2025

**Цель работы:**

Научиться использовать библиотеку face\_recognition для сравнения лиц на двух изображениях и определения, является ли лицо на неизвестном изображении тем же самым, что и на известном.

***Задание***

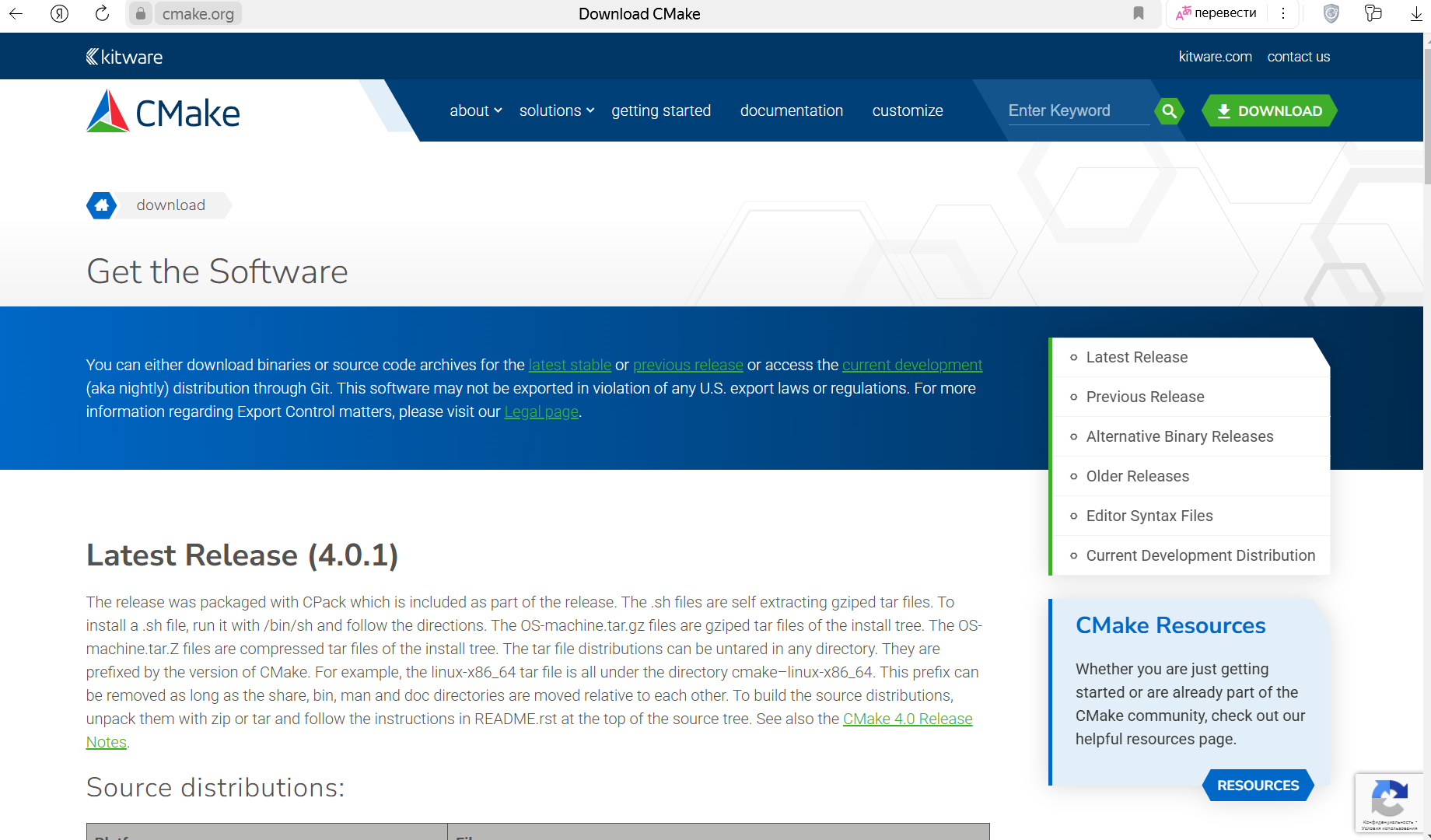
* Используйте предоставленный код для загрузки и кодирования известного и неизвестного изображений.
* Сравните кодировки лиц и определите, присутствует ли лицо известного человека на неизвестном изображении.
* Проанализируйте результат сравнения. Объясните, что означает результат True или False возвращённый функцией compare\_faces.

Выполнение:

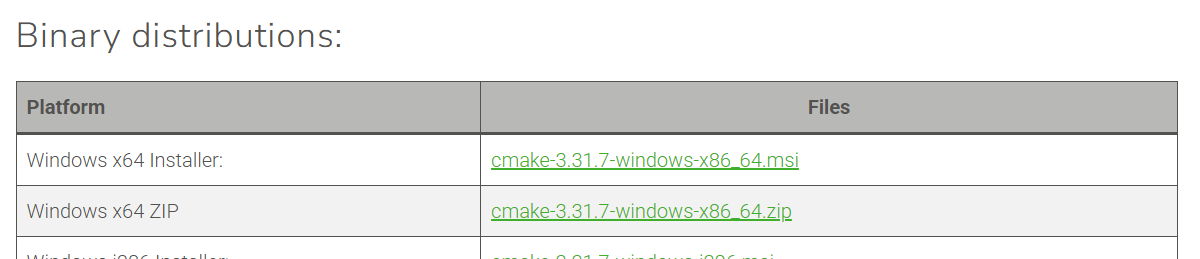
Шаг 1: «Установка библиотеки».

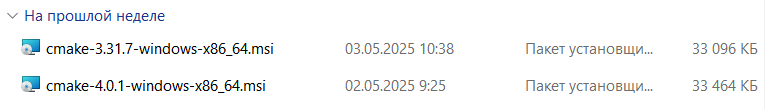
Чтобы всё работало корректно, пришлось разбирать ошибки кода, которые выдавала программа. Выяснилось, что необходимо установить сторонние библиотеки, которые помогают работе face\_recognition.

Прежде всего - это библиотека cmake. Она была скачана с официального сайта, имя которого было обозначено в ошибке при первых попытках установки.

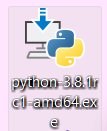


Была установлена версия 3.31.7, потому что при работе с более поздними возникали ошибки.

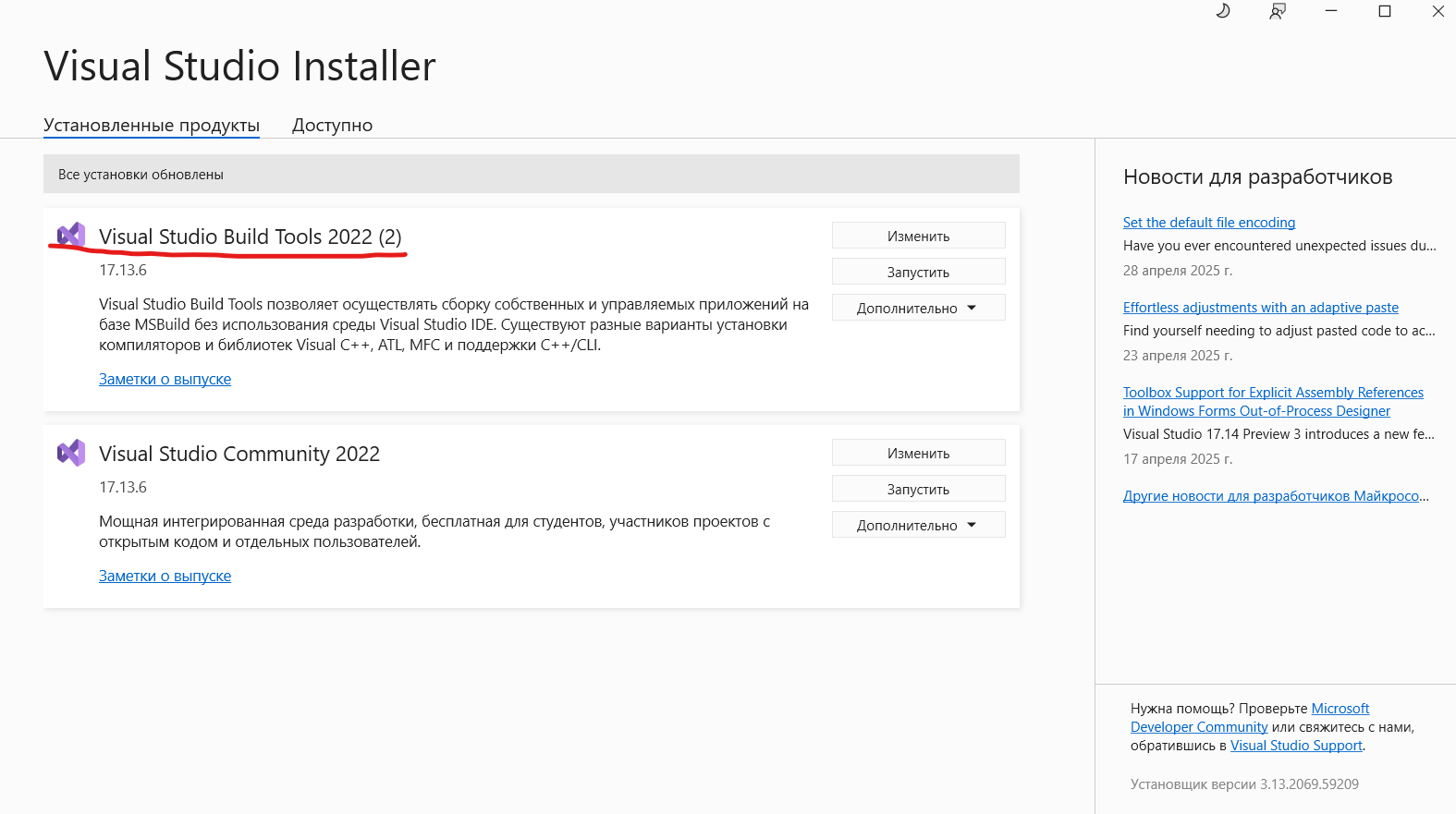
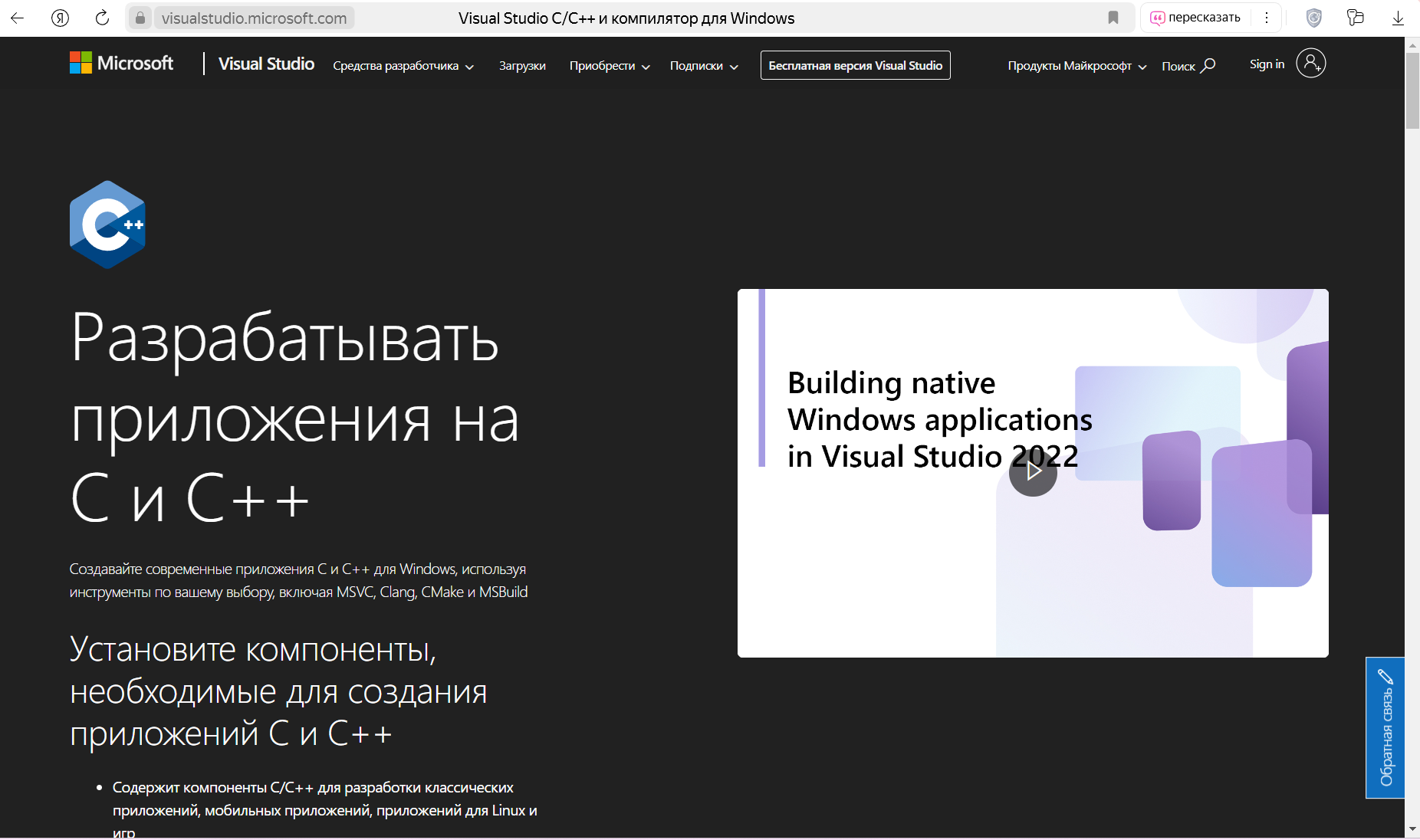




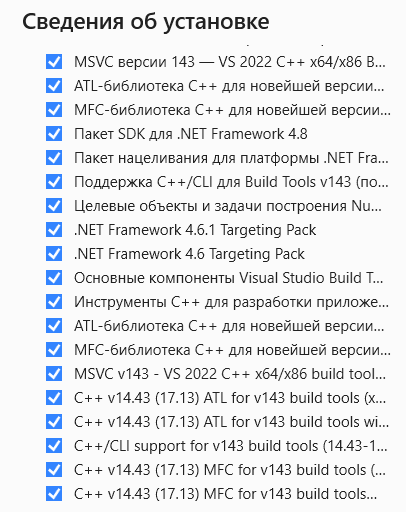
После нужно было установить библиотеку dlib, но была проблема с совместимостью версий. На ноутбуке стояла версия 3.12.10, поэтому потребовалось установить более раннюю версию - 3.8.1rc1. Она понадобилась после загрузки всех библиотек для импорта и использования.

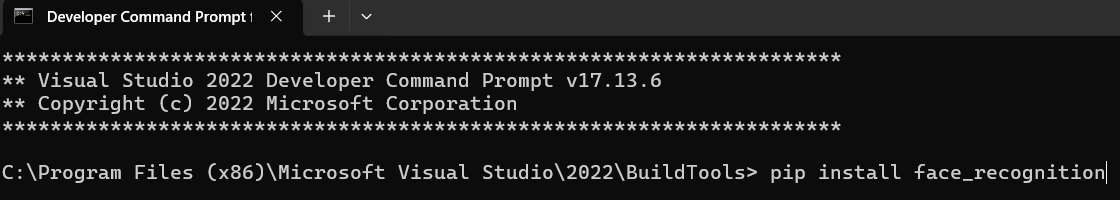


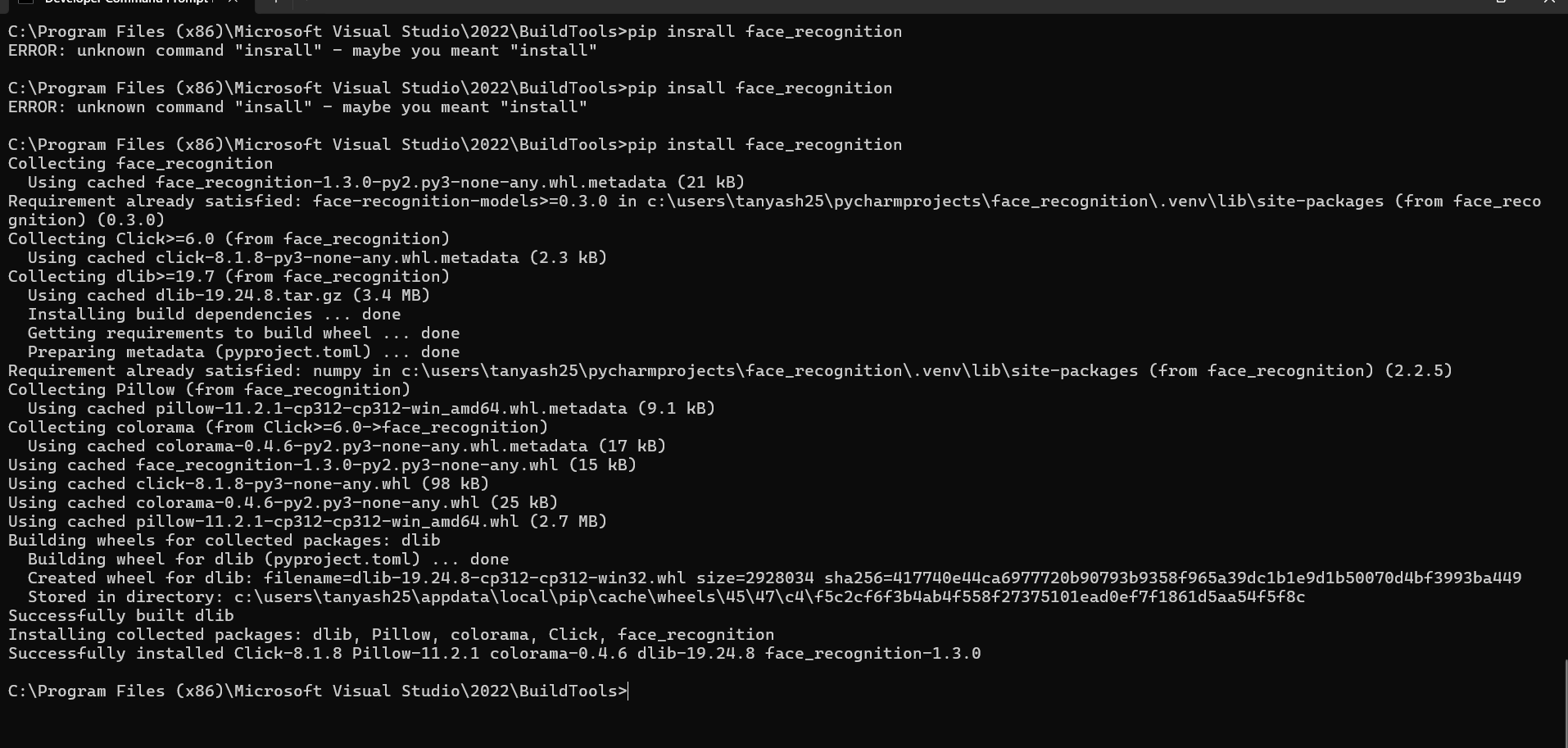
Самой последней проблемой, которую предстояло решить, была проблема с компилятором для языка C/C++. Удалось это решить путём скачивания дополнения для Visual Studio:



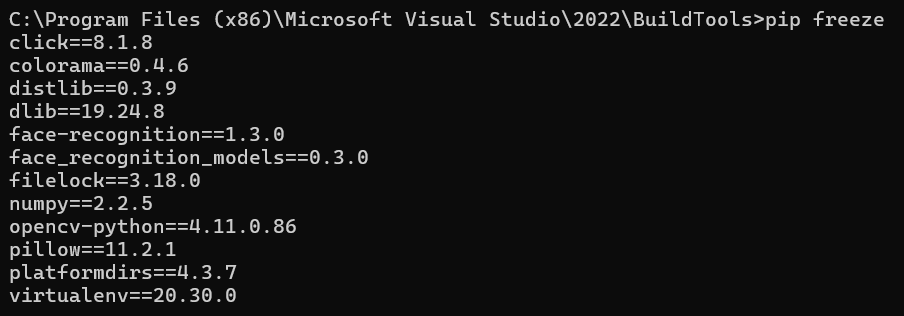
Установив галочки на все инструменты для C++ build tools и скачав нужные компоненты, получаем возможность для скачивания face\_recognition.



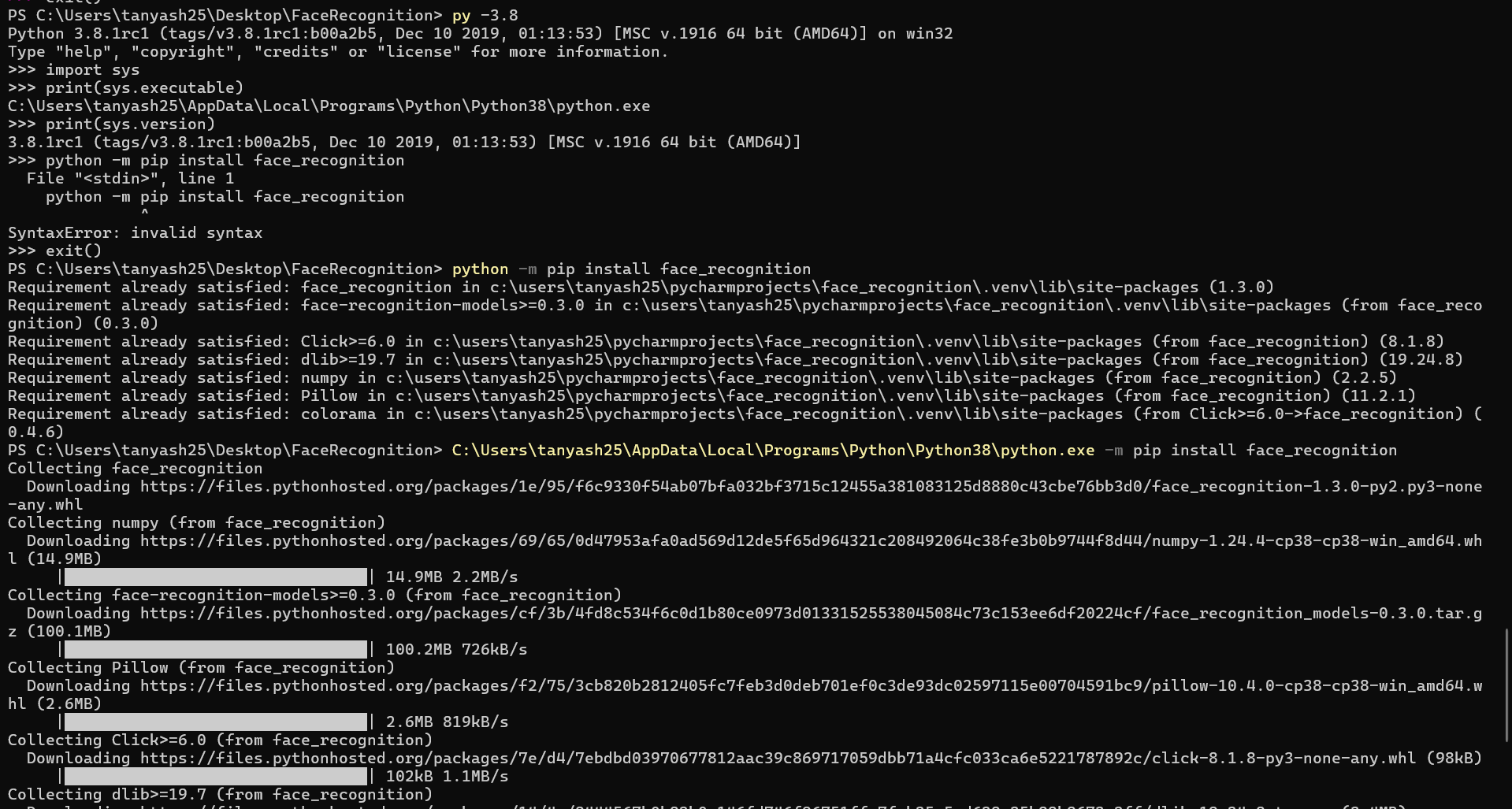


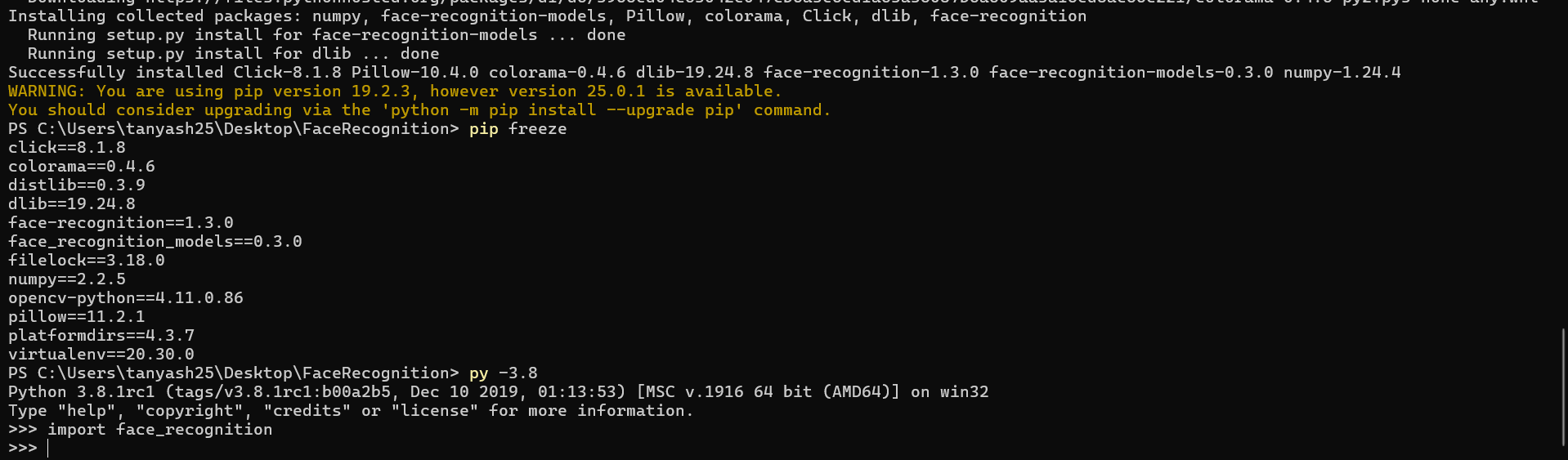


Успешно установлены все необходимые библиотеки.



Чтобы это работало в версии Python 3.8.1rc1, окружение было повторно установлено в соответствующую директорию:

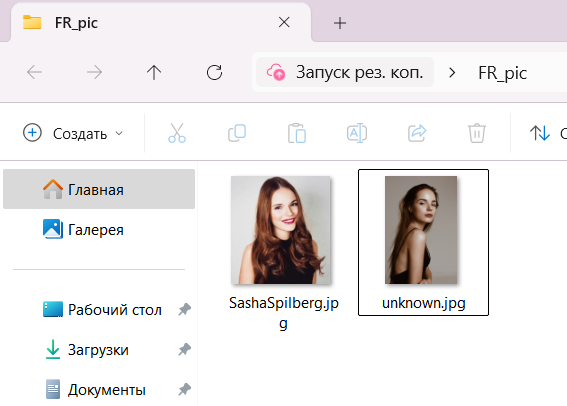




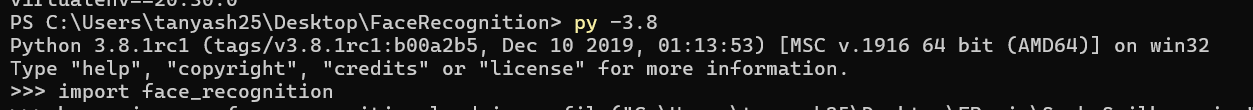
После этих действий удалось импортировать библиотеку в python.

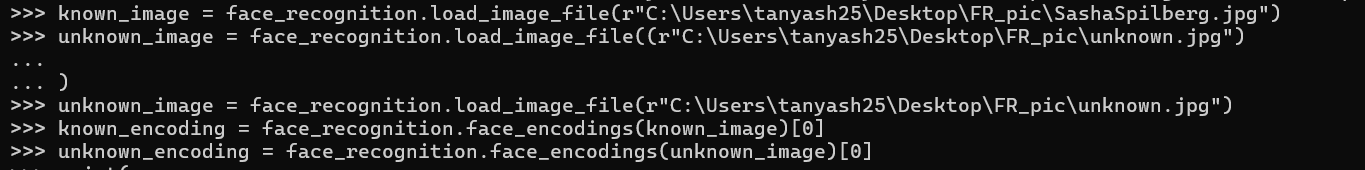
Шаг 2: «Использование библиотеки в python»

Теперь используем подготовленные заранее изображения из папки.



Запишем и выполним код в консоли:





Так как python воспринимает обратные слеши в строке как элементы «управляющих символов», нужно перед строкой поставить r. После этого строка будет восприниматься нормально, а возможное появление управляющих символов будет игнорироваться.

Небольшое пояснение:

face\_recognition.face\_encodings(known\_image)

Эта функция берет массив NumPy, представляющий изображение (known\_image), в качестве аргумента.

Она находит все лица на изображении.

Для каждого найденного лица она вычисляет “кодировку лица” (face encoding). Кодировка лица - это 128-мерный вектор (массив из 128 чисел с плавающей точкой), который представляет уникальные черты лица. Этот вектор генерируется с использованием глубокой нейронной сети (CNN).

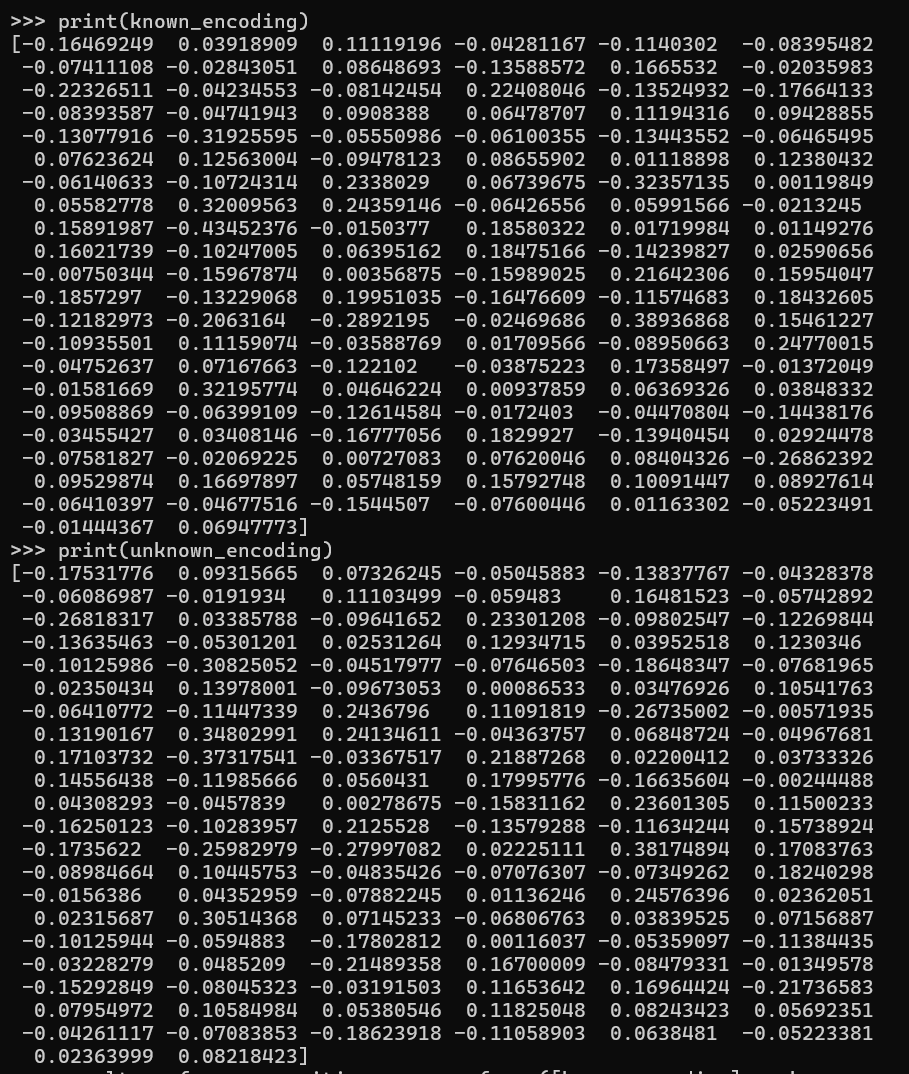
Она возвращает список кодировок лиц. Если на изображении нет лиц, возвращается пустой список. Если на изображении несколько лиц, возвращается список, содержащий кодировку для каждого лица.

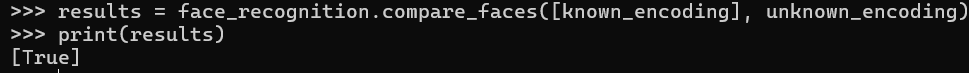
Тип переменной: list. Список numpy.ndarray.

Содержание переменной: Список массивов NumPy. Каждый массив NumPy имеет размерность (128,) и представляет кодировку лица.

Шаг 3: «Сравнение лиц на фотографиях»

Используя print(), можно увидеть, что содержится в переменных с припиской \_encoding:



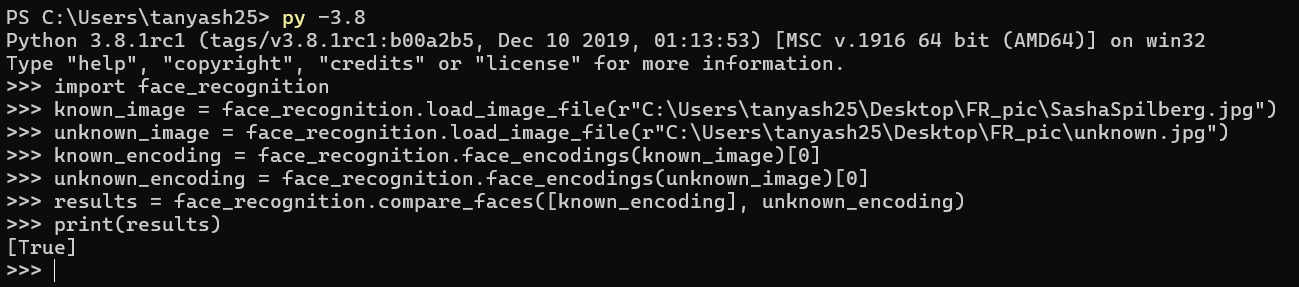


При получении ответа «True», можно судить о следующем:

Функция определила, что лицо на “неизвестном” изображении (unknown.jpg) с большой вероятностью является тем же самым человеком, что и лицо на “известном” изображении (known.jpg). Алгоритм пришёл к такому выводу на основе сравнения 128-мерных кодировок лиц, полученных с обоих изображений.

И это действительно так, фотографии были подобраны таким образом, чтобы на них был один и тот же человек.

Полный код программы (без просмотра содержимого encoding):



Текстом:

import face\_recognition

known\_image = face\_recognition.load\_image\_file(r"C:\Users\tanyash25\Desktop\FR\_pic\SashaSpilberg.jpg")

unknown\_image = face\_recognition.load\_image\_file(r"C:\Users\tanyash25\Desktop\FR\_pic\unknown.jpg")

known\_encoding = face\_recognition.face\_encodings(known\_image)[0]

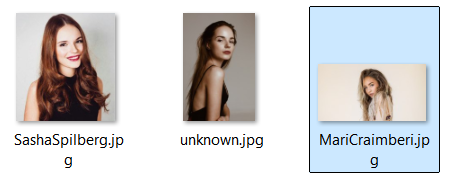
unknown\_encoding = face\_recognition.face\_encodings(unknown\_image)[0]

results = face\_recognition.compare\_faces([known\_encoding], unknown\_encoding)

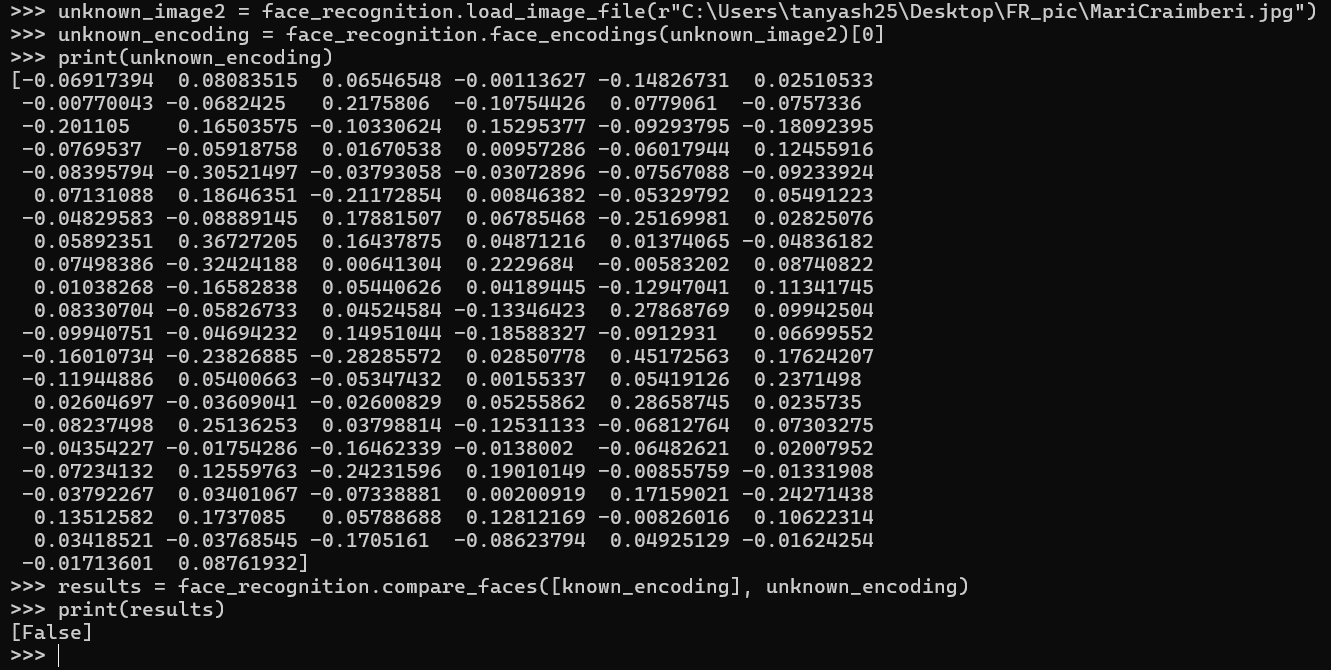
print(results)

Шаг 4: «Другой человек»

Для того, чтобы лучше погрузиться в работу библиотеки, был сделан ещё один тест - в этот раз с фотографией другого человека:

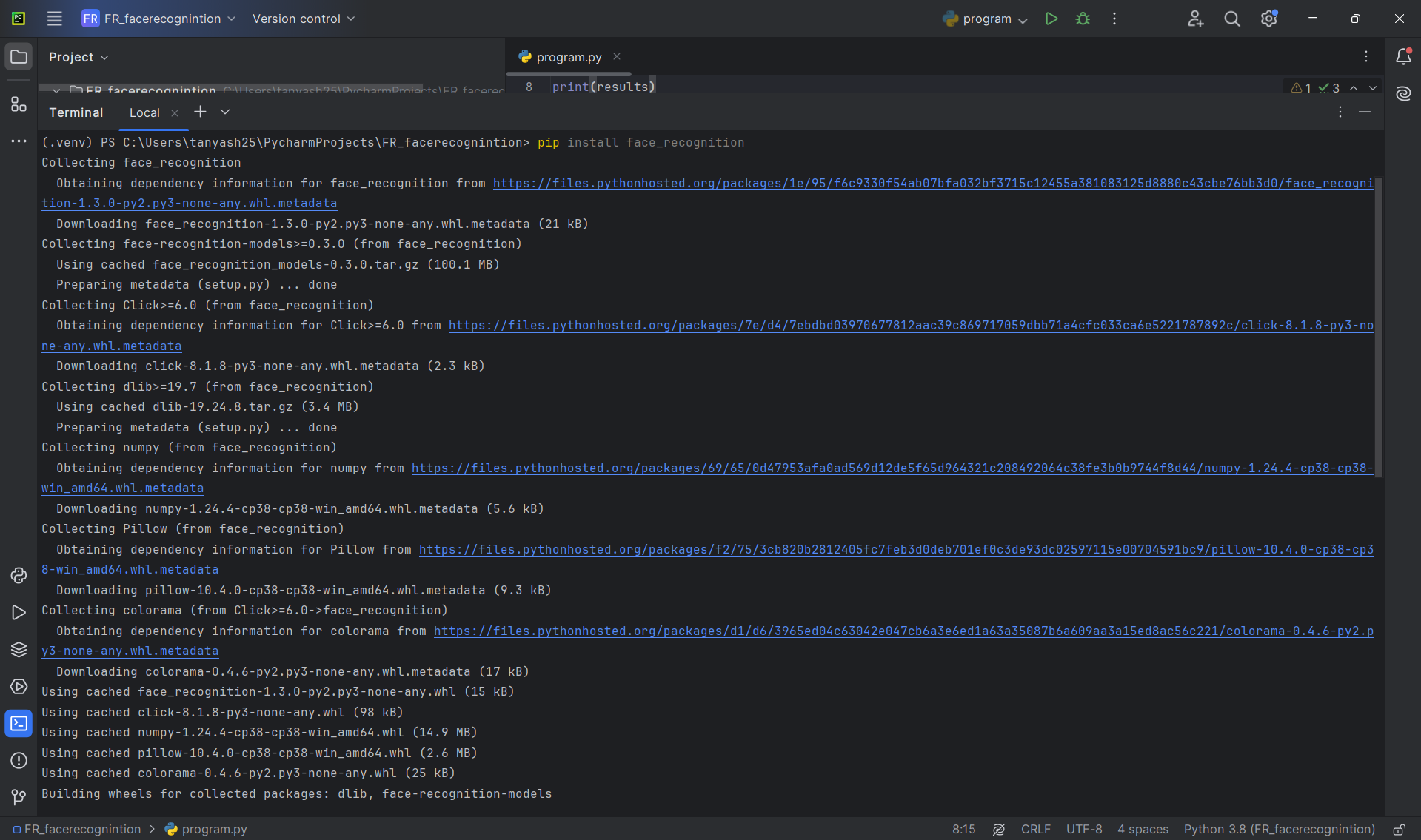


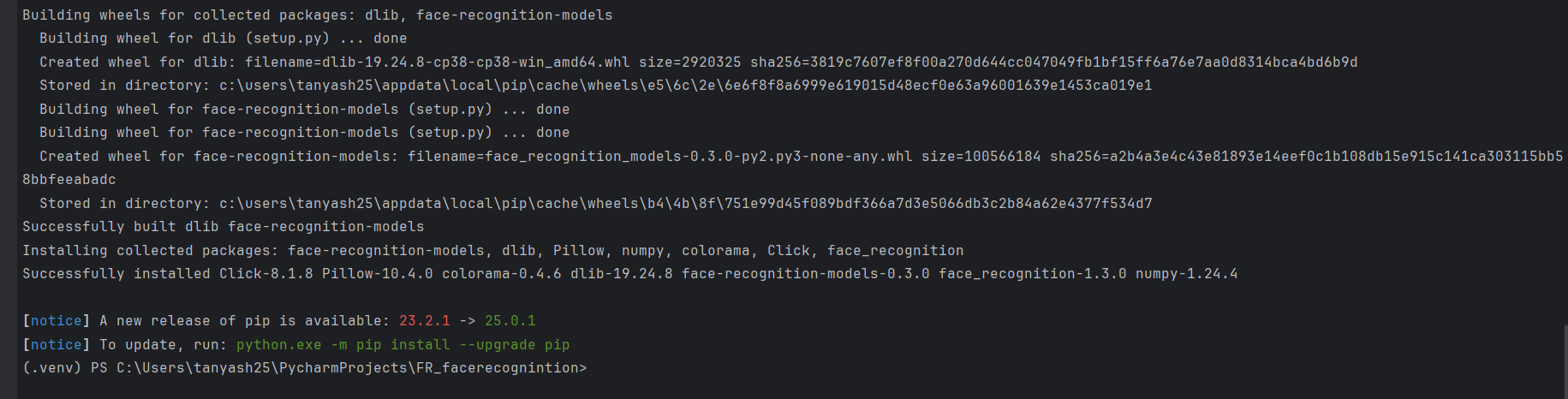
Коды:



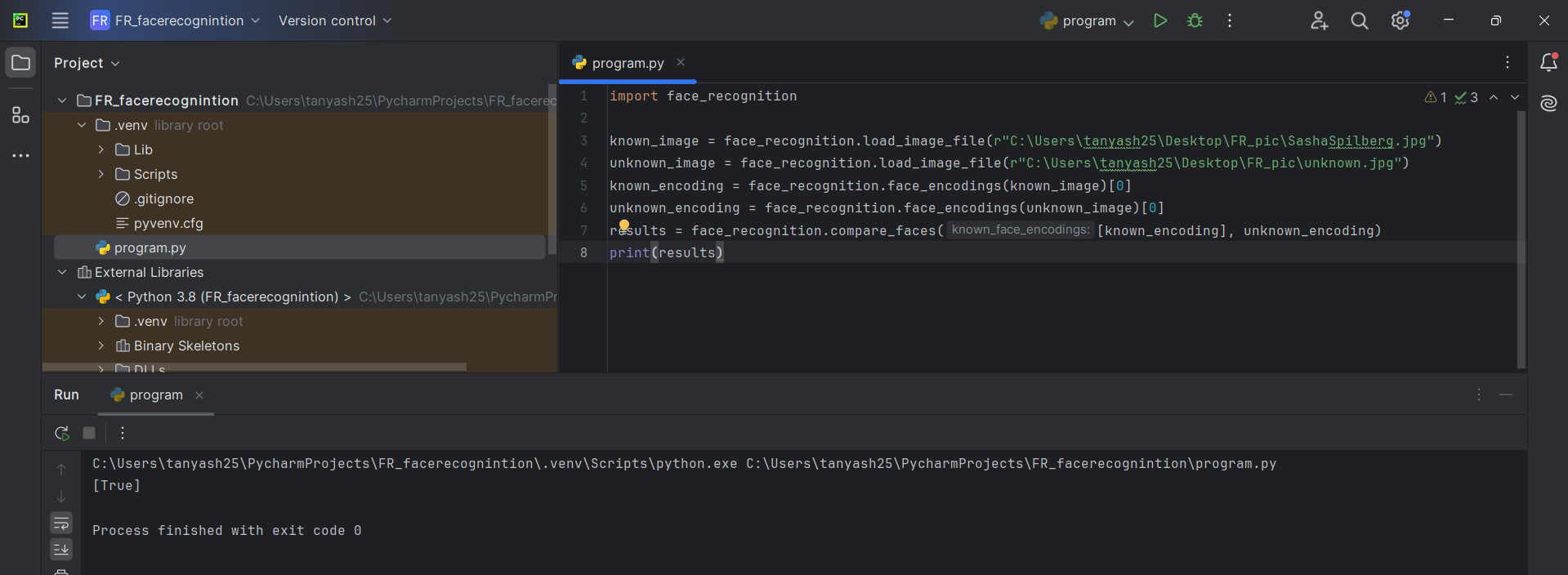
Результат получился отрицательным, что является верным прогнозом.

Шаг 5: Установка в PyCharm





Тестирование кода:



Пусть и с задержкой, получается тот же результат, что и в консоли.

**Вывод**: В ходе данной лабораторной работы был получен навык использования библиотеки face\_recognition, установки виртуального окружения по требованиям, работы с разными версиями Python через консоль и в среде разработки PyCharm. После сравнения двух лиц на изображениях получилось определить, что лицо на неизвестном изображении является тем же самым, что и на известном. Дополнительно была проведена проверка с изображением другого человека. Результат получился отрицательным, что совпадает с действительностью.

## Ответы на вопросы: